SITUS LOGIC

Design Tools for Integrated Asynchronous Electronic Circuits

19 June 2003

Sponsored by
Defense Advanced Research Projects Agency
Microsystems Technology Office
ARPA Order K476/66

Issued by U.S. Army Aviation and Missile Command Under
Contract No. DAAH01-03-C-R021

Situs-TR-03-03

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	23 JUNE 2003	FINAL 24OCT02 – 24JUN03

4. TITLE AND SUBTITLE
CAST: A SUITE of CAD TOOLS for ASYNCHRONOUS VLSI

5. FUNDING NUMBERS
DAAH01-03-C-R021

6. AUTHOR(S)
ALAIN J. MARTIN, MIKA NYSTRÖM, CATHERINE WONG

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
SITUS LOGIC
1442 LOMITA DRIVE
PASADENA, CA 91106

8. PERFORMING ORGANIZATION REPORT NUMBER
Situs-TR-03-03

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
DARPA-MTO, 3701 N. Fairfax Drive, Arlington VA 22203-1714
US. ARMY Aviation & Missile Command
ATTN: AMSAM-RD-WS-DP-SB
Bldg 7804, Rm 212, Redstone Arsenal, AL 35898

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
—

11. SUPPLEMENTARY NOTES
—

12a. DISTRIBUTION/AVAILABILITY STATEMENT
UNLIMITED

12b. DISTRIBUTION CODE
—

13. ABSTRACT *(Maximum 200 words)*

The objective of this Phase I study was to demonstrate the feasibility of a suite of industrial CAD tools for the design of high-performance, energy-efficient, asynchronous (clockless) VLSI integrated circuits, based on the technology invented at Caltech.

Situs Logic will develop and commercialize a complete suite of tools including logic synthesis, physical synthesis (layout synthesis) analysis, simulation, verification, and transistor sizing. The main technical activities and developments were the design of a prototype toolset to test and demonstrate the approach. The design flow is not entirely automated. But the results are promising: the circuits designed with the tool are within the state of the art.

14. SUBJECT TERMS	15. NUMBER OF PAGES
CAD tools, EDA tools, asynchronous VLSI, QDI, design tools, logic synthesis, HDL, CHP, layout, simulatio	15
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	

Design Tools for Integrated Asynchronous Electronic Circuits

19 June 2003

Sponsored by
Defense Advanced Research Projects Agency
Microsystems Technology Office
ARPA Order K476/66

Issued by U.S. Army Aviation and Missile Command Under
Contract No. DAAH01-03-C-R021

CAST: A Suite of CAD Tools for Asynchronous VLSI

Alain J. Martin, PI
Situs Logic
1442, Lomita Drive
Pasadena, CA 91106
(626) 799-7830

19 June 2003

Effective date of Contract: 24 October 2002
Reporting period: Final
Contract Expiration Date: 24 June 2003

1. Summary

The objective of this Phase-I study was to demonstrate the feasibility of a suite of industrial CAD tools for the design of high-performance, energy-efficient, asynchronous VLSI circuits based on the Caltech technology. Situs Logic's general strategy in the EDA-tools market is to develop and commercialize a complete suite of CAD tools for the design of asynchronous, QDI, VLSI systems including synthesis, analysis, simulation, verification, at the logical and physical levels.

Situs has developed a business model for the commercialization of the CAD tools, and has designed the prototype of the tool suite based on this business model and the Caltech approach.

The market for asynchronous tools will not be at first the mainstream market, but rather some "early adopters" designing low-volume high-profit chips, for instance for defense or space applications. Inside the market segment of asynchronous VLSI tools, the competitive advantage pursued by Situs is differentiation rather than cost leadership, even though the Situs tools will be priced significantly below equivalent tools in the mainstream EDA market.

The main technical activities and developments were directed towards the design of a prototype tool-suite to test and demonstrate the feasibility of the approach. Although the design flow is not entirely automated yet—some steps still have to be preformed manually—the results of the effort are very promising: The circuits synthesized with the prototype tool are easily competitive with the state of the art.

The main difficulty for an inexperienced designer of asynchronous circuits is at the logic-synthesis level since the absence of clock makes the logic synthesis of asynchronous circuits very different. The Situs tools will hide this difficulty by having the core part of the logic synthesis entirely automated. Another difficult part of the synthesis is the decomposition of a large original HDL description into a collection of modules amenable to hardware implementation. This procedure will also be entirely automated and, combined with a powerful optimization step, will produce better solutions than what most designers can do by hand.

The different programs of the suite will be structured so as to make it possible for the designers to tailor the tools to their styles, needs, and experiences, by carefully designing the interfaces in such a way that replacing one tool with another should be easy.

Another major concern is that it doesn't do much good for the tools to allow designers to synthesize 95% of their design if they get stuck on the remaining 5%. In Phase II, the tools will be augmented with an extensive *tool-kit* that will provide standard solutions for most (if not all) the unusual technical problems that the tools cannot or should not solve.

For high-level description, the Situs tools will use the language CHP, wich was developed and refined at Caltech over the last decade, rather than VHDL or Verilog. However, an alternative will be provided in the form of a subset of VHDL—and later the same for Verilog—that exactly implements the constructs of CHP. This language is called CHDL. A user designing in CHDL will be in the familiar environment of VHDL. The resulting code will be translated in CHP at little cost. From then on, the design flow will be the same.

The general framework of the Situs tool suite has been thoroughly investigated and defined. The interfaces between tools and the declarative language *Cast*, which ties the different representations of a design together, are being designed in such a way as to allow the designer to customize the tools.

The effort expended so far can be summarized as follows. At the logic-synthesis level, an automatic procedure exists to decompose any given CHP or CHDL program into a network of small components. Also, the core of the logic synthesis—the transformation from CHP to PRS—has been formalized. At the simulation level, the framework of the Situs toolset (*Cast*) has been defined and the interfaces have been delineated. A CHP simulator now exists, and a new method for mixed-level simulation or cosimulation has been defined.

At the physical-design level, a standard-cell library has been defined and built. The front-end of a cell generator (*stackgen*) has been developed, as well as a preliminary placer and router. An extended version of the PRS language, called XPRS, has been defined that contains information about transistor ordering and sizing. XPRS makes it possible to layout a chip without need to edit the layout manually.

2. Body of Report

2.1. Milestone/Task Status

In this section, the status of the project is reported and compared against the baseline. The program is on schedule. Priority was momentarily given to the physical design part of the project to support the Caltech PACC program which is in the physical-layout phase, and to use a real design as testbed for the tool development. The emphasis has now returned to the logic synthesis, which is the most innovative part of the effort.

The effort expended on each task is as follows. At the logic synthesis level, a new representation, CAST03, has been designed to tie together the different representations of a design and the different hierarchies used to structure a complex design. Also, the design of a tool implementing the DDD decomposition procedure for logic synthesis is well under way. At the physical-design level (the Situs back-end) the transformation from production-rule representation to layout is more than two-thirds automated. A new production-rule representation, XPRS, has been defined that contains information about transistor ordering and sizing. XPRS is used to generate layout. A proprietary router has been developed, together with tools for cell generation and placement.

CHDL, a subset of VHDL implementing the constructs of CHP, as an alternative high-level language has been defined and developed: Designers used to VHDL (or required to write in VHDL) can now use CHDL as a high-level language. The CHDL code is automatically translated into CHP.

The next section gives a completed narrative of the different tasks mentioned above, together with the outstanding problems, and new problem areas.

3. Narrative

3.1. Overview

The Situs toolset consists of high-level synthesis and decomposition tools that manipulate high-level CHP descriptions as well as low-level tools for translation of production-rule sets into layout, placement, and routing. The Situs toolset provides several alternative paths for designers: two alternatives are shown in the following figure—the standard Situs solution (described in more detail below) and a different path, called syntax-directed compilation.

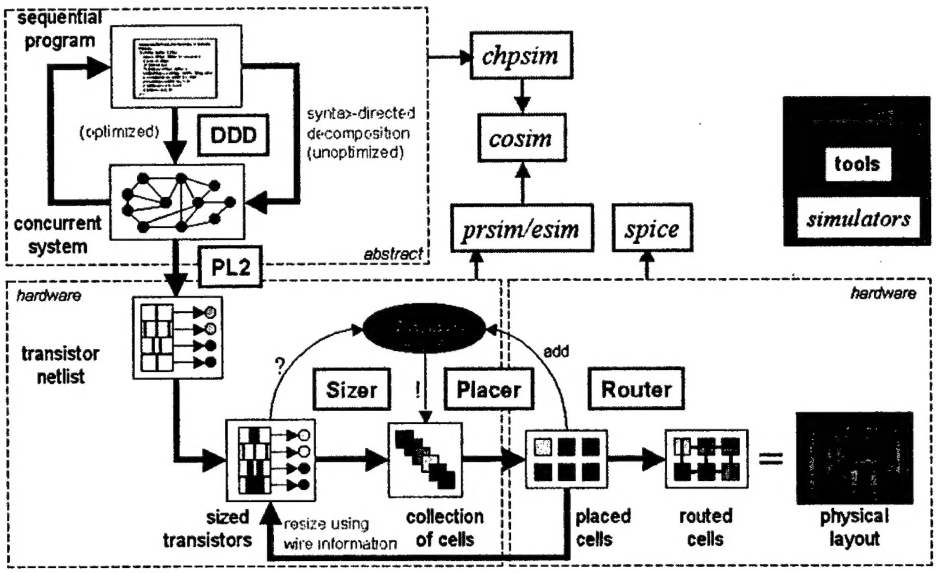

Figure 1: The Situs Toolset

3.2. The Situs Framework

The biggest challenge in EDA tools is that posed by the need to manage very large designs. Every year it is the same story: designers are faced with designing tomorrow's systems with today's or yesterday's systems. And designs are getting ever larger; the day when a designer could have an entire system in his head at one time is long gone. The only reasonable way of handling the design process for large systems is to *subdivide* the process, both horizontally (by decomposing a large system into smaller, independent subsystems) and vertically (by adding more, smaller steps to the design process so that at each step only a limited amount of information needs to be in the designer's or design tool's memory). The catchword for this sort of horizontal/vertical decomposition is *hierarchy*: a design can be thought of as having levels of hierarchy with more and smaller pieces in each lower level of the hierarchy.

The main promise of asynchronous design techniques is that they permit designers to think of their systems in a hierarchical sense. This is not so easy in clocked environments, where the ubiquitous presence of the clock breaks the hierarchy and makes it necessary to perform key design and analysis steps on a flattened representation. Moving away from flattened representations

and keeping all aspects of the design hierarchical is necessary to realize the complete potential of asynchronous design techniques. It is not surprising that existing design systems, which are optimized for representing clocked designs, are inadequate for supporting the asynchronous design revolution.

3.2.1. Situs's design hierarchies

The Situs asynchronous toolset is built around a way of designing circuits that consists of a sequence of systematic, provably correct transformations that take the designer from an initial specification to a final layout geometry. This sequence of transformations passes through a number of representations, e.g., CHP, HSE, PRS, transistor netlists, and various representations of layout geometries. Situs's own tools support several transformation paths, and users are free to add their own, using their own tools or standard commercial tools from other commercial CAD vendors. This sequence of transformations is the entire design process: by recording the sequence of transformations, one records the design process, and one can use this record to audit it. For instance, hardware verification is the act of verifying that the representations at different stages in the design process are equivalent—for instance, when designing a microprocessor, a designer would want to check that the transistor circuits in fact compute the logic functions he intended, which in turn should implement the desired RTL, and so on. In the Situs flow, the design task can be subdivided into smaller design tasks or continued in a sequence of steps going from the first, high-level specification to the final GDS-II polygon representation of the layout. The design process can thus be organized as a *design hierarchy.*

One of the basic transformations performed on a hardware specification is that of *refinement.* Refinement is the act of replacing a specification with an implementation, and it involves moving from a general specification to a specific implementation and from a high-level description to a more detailed description. For instance, the specification "an adder" can be refined to "an eight-bit adder" and further to "an eight-bit ripple-carry adder." Refinement is well-known from the software-engineering world, where the inheritance encountered in object-oriented programming is a kind of refinement. A large design goes through many levels of refinement, and the relationships between the different levels is hierarchical—perhaps the ALU is refined into something containing an adder and a multiplier, and after that the adder is refined separately from the multiplier. Refinement thus induces a *refinement hierarchy* on a design. The refinement hierarchy is more specific than the design hierar-

chy: whereas the design hierarchy simply ties together the steps taken by the system designer, in whatever way the system designer may have derived the pieces, the objects in the refinement hierarchy, such as the adders mentioned above, obey formal rules that enable them to be substituted for each other in particular ways. The objects in the design hierarchy do not necessarily obey such rules; or if they do, the rules may not be practically expressible: there is for instance no clear inheritance relationship between an ISA specification and the specification of an adder used to build a system satisfying the ISA specification.

Modern hardware systems are very large and getting larger. It has become impossible for designers to think of a modern hardware system as a single circuit; there is simply too much detail. Instead, a hierarchical representation is usually used. For instance, a microprocessor consists of several execution units, an instruction-fetch unit, an instruction cache, and so on; an instruction-fetch unit consists of a few adders plus control circuitry, and so on. This hierarchy, the *instantiation hierarchy,* is nested many levels deep. The instantiation hierarchy is a concrete, essential part of the representation of a hardware system, and it is important for design efficiency that tools be able to handle it smoothly.

3.2.2. The CAST03 Language

Keeping track of the three hierarchies mentioned above—design, refinement, and instantiation—is one of the main tasks of the Situs toolset. The part of the system that does this is called CAST (the acronym originally stood for "Caltech Asynchronous Synthesis Tools"). Several versions of CAST have already been developed, most of them at Caltech. Each one of these versions has represented a great advance over previous CAST systems, and they have been used in the design of a large family of asynchronous chips, including the early Caltech microprocessor and the Caltech MiniMIPS, as well as the still larger chips designed at Fulcrum Microsystems, Inc. However, none of the existing CAST systems has yet fulfilled the goal of properly tracking the three hierarchies. A new CAST system that will correct this deficiency, CAST03, is part of the Situs toolset.

The CAST03 system is inspired by prior work in hardware-representation systems and notations. The main design issues in a hardware-representation system are convenience, efficiency, and soundness. Convenience is of course a major issue in a system that will be used by human designers; the kinds of conveniences that users want in a hardware-representation system include

syntactic convenience (e.g., concise notation) and a high degree of parameterizability. Since hardware systems are extremely complex (more so the closer one gets to the layout representation), efficiency is also a major issue; it is widely accepted by VLSI designers that their tools will only run on large, expensive server computers with the maximum memory installed. Finally, soundness is extremely important: soundness refers to the degree to which the design transformations can be tracked and audited and the degree to which different levels of specification can be verified against each other. A high level of soundness is necessary for the design advantages of asynchronous design to be maximized. It is no surprise that the three goals of convenience, efficiency, and soundness are generally at odds with each other.

No existing design environment simultaneously provides the level of convenience, efficiency, and soundness that asynchronous design techniques enable. This makes CAST03 a key part of the Situs tool suite. CAST03 unites the goals by making judicious choices of features from existing design systems, and while the features of CAST03 are inspired by a variety of existing design systems, both academic and industrial, no existing design environment can describe asynchronous circuits as gracefully as Situs's.

3.3. The Situs Synthesis Tools

3.3.1. High-Level Synthesis

The first step in the Situs design flow is process decomposition, which transforms sequential high-level descriptions of circuit behaviour into a system of communicating modules. Each module is still expressed in a high-level language and can be individually synthesized at lower levels. The goals of process decomposition are to expose concurrency and facilitate low-level synthesis while producing a system with an acceptable throughput but not a surfeit of communications. (In most QDI systems, the computation of values consumes significantly less energy than the communication of these values.)

Previous approaches to automated process decomposition have been syntax-directed and unable to produce modules small enough to be implemented as the fine-grain pipeline stages (*precharge half-buffers*, or *PCHBs*) used in the high-performance MiniMIPS and Lutonium asynchronous microprocessors. The Situs tool flow features *data-driven decomposition (DDD)*, the first decomposition method to target the fast PCHB asynchronous circuit family. Before DDD, designers wishing to create high-throughput asynchronous systems could spend weeks creating an energy-efficient decomposition to meet the desired throughput, and even after that effort, the results would vary

greatly depending designer experience and intuition. The decision to make all intermediate stages in the Situs tool chain human-readable gives users the freedom to choose between DDD, syntax-directed decomposition, or performing the process decomposition by hand. The remainder of this section focuses on the algorithms and tools for DDD.

Data-Driven Decomposition

DDD performs data-dependency analysis on the sequential CHP and then decomposes the program into a system containing one module for each variable. The modules contain both the computation of the variable (encapsulating all assignments to that variable in the original program) and new intermodule communications which read in values of other variables and send out the newly computed value. (External output channels that appear in the original sequential CHP are considered special variables and are decomposed into their own modules.) By considering the flow of data instead of the syntax of the original program, DDD eliminates unnecessary synchronization in the system.

When this analysis and decomposition is performed in isolation, the new modules are not guaranteed to fit into the CHP template for PCHB circuits. DDD decomposes any deterministic CHP program into a system of PCHB-implementable modules by first rewriting the sequential program so that every variable is assigned a value at most once during the execution of an iteration of the main loop. This transformation to *dynamic single assignment (DSA)* form may involve splitting variables into new ones, eliminating inner loops, and adding assignments to the ends of guarded commands (i.e., selection statement branches).

After DSA conversion, data-dependency analysis and decomposition, DDD has created a working concurrent system where every module can be implemented by a PCHB circuit. The decomposition may have gone too far though—modules may be smaller than is necessary to fit the physical and performance constraints placed on the size of PCHBs. The last stage of DDD is therefore to cluster the DSA modules into larger modules to improve energy (reducing the number of communications in the system) and performance (cutting forward latency while still running at the desired throughput). Decomposition is followed by some recomposition so that the user can prioritize which metrics are most important in terms of the performance of the final system.

Clustering is implemented in two phases. In the first phase, DSA modules along the critical path of a system are repeatedly clustered in series until they are too large to implemented as single PCHB circuits. The second phase employs a global optimization heuristic to cluster modules in parallel while adding slack-matching buffers to keep the entire system running at the desired throughput. The final output is a concurrent system where communicating modules may implement the computations of one or multiple variables, or may be simple buffers inserted to improve system performance. All modules fit the PCHB circuit template.

Tools

In Phase I, we created tools that convert deterministic CHP programs into DSA form, analyze the data dependencies in a DSA program and output a system of CHP modules equivalent to the original sequential specification. These tools implement the basic algorithms comprising the first three stages of DDD, but do not incorporate all of the features envisioned for the final tool. For example, input CHP cannot yet contain nested loops (which, actually, are rare in practical circuits), and conditionality is not yet allowed in most intermodule communications introduced by decomposition. These tools have been successfully tested on various programs, including the complex Instruction Fetch program used in the Lutonium (asynchronous 8051 microcontroller). A tool that implements the final clustering stage of DDD is currently in development.

3.3.2. The Situs Back-End: Production Rules to Layout

The Situs toolset takes the Production Rule Set (PRS) as the boundary between logical and physical design; production rules are the target of the logical design and the specification for the physical design. The physical design in the Situs flow is different in many ways from a traditional synchronous physical-design flow: many optimizations are possible at the circuit level, and even at the layout level, that depend on information from the logical-design level. For instance, facts such as "nodes x and y can never both be **true** at the same time" can be used in order to perform circuit and layout optimizations. A tighter integration of the tools than is seen in standard tool flows is therefore desirable, and the Situs physical tools provide this.

Low-level production-rule representation: XPRS

The physical tools start with a production-rule set. This is a purely logi-

cal specification, and it is not sufficient for driving the physical design tools. In order to provide a specific enough (but not over-specific) description of the system, the Situs toolset uses a new representation called the Extended–Production-Rule Set (XPRS). This representation specifies transistor-gate ordering and transistor-gate widths, but it does not specify the complete circuit topology, nor does it specify any other geometry information. The XPRS notation is ideal for transistor sizing and it is also ideal for human-produced low-level descriptions: using XPRS, a designer can specify all relevant details about an asynchronous circuit implementation without having to edit actual chip layout directly.

The introduction of XPRS subdivides what was formerly one task (sizing, gate ordering, and specification of the netlist) into two tasks (sizing and gate ordering on the one hand and specification of the netlist on the other). This is in line with the Situs philosophy that each design tool be simple and interchangeable and that it can be overridden by human input.

Because XPRS is used for sizing, the Situs tools convert the standard PRS into XPRS as the first step in the physical-design flow. This conversion is done in one of three ways:

- Automatically through gate matching.
- Automatically through XPRS generation from PRS.
- Manually or by logical-design tools.

The three ways are used as follows: the automatic methods are used when a PRS is given. First of all, gate matching is performed: the Situs system is able to match a given PRS against a gate library whose cells are described in XPRS—in this case, the presence of a cell in the gate library is taken to mean that the transistor-gate ordering is arbitrary, and logically equivalent cells are matched against the given PRS. Secondly, remaining PRS are converted by a special XPRS generator: this generator makes the decisions regarding transistor-gate ordering and gate sharing; this is the least preferred approach because the XPRS generator has to be conservative about its designs in order to guarantee that they function properly. The final method of generating XPRS is the simplest: the user simply specifies the gate ordering. Normally, however, the "user" is a higher-level tool in the Situs logical-design suite; this tool will have the necessary information to pick a reasonable gate ordering and sharing.

Placement and routing

Placement and routing are the final steps in the physical-design flow. The Situs toolchain is extremely flexible with regard to cell placement. Cells can either be placed manually by the designer by leaving the appropriate directives in the CAST03 code, or the placement can be done automatically. If it is done automatically, special directives can still be used in order to perform datapath placement—the regularity of a datapath means that extra information is available in order to optimize the routes. The CAST03 system makes it easy for the designer to specify this extra information.

Routing is performed by a proprietary Situs router. The Situs router routes nets one at a time, it supports rip-up-and-reroute for batch mode "hands-off" routing, and it works with all standard ASIC processes. It furthermore incorporates an efficient approximate Steiner-tree router (based on minimum spanning trees) for routing nets that connect more than two points together, which is important for asynchronous circuits because it is not unusual for circuit nodes fan out to several dozen transistor gates. The Situs router is unique in that it can easily handle hand-drawn custom layout without special directives telling it where to connect to the layout. The Situs router is also "opportunist" router that will connect anywhere it can on a net. This improves routability because the router is able to connect to the layout in more places than a standard router could, and more importantly, it makes it very easy to combine standard cells with hand-drawn layout, in keeping with the Situs designer-assisted compilation philosophy.

3.4. Future Plans

Phase I was used to design a prototype of an EDA toolset for asynchronous VLSI design based on the Caltech methodology, and to define a product from a business and marketing point of view. The toolset can already be demonstrated on a reasonably large part of a system with excellent results compared to the state of the art.

In Phase II, the prototype will be turned into an industrial-strength toolset. An α-version will be ready by the end of the first year for use by a small group of early adopters, and a β-version by the end of Phase II. Situs is eager to put the tools in the hands of industrial users outside of the async research community as soon as possible to get feedback from them, as it is difficult for the developers to foresee what will be the main stumbling blocks for such users.

A new transistor-sizing algorithm has been developed at Caltech, which

Situs will implement and integrate into the tool suite in Phase II. The optimization metric used by the algorithm is Et^n where n can be chosen by the designer so as to fix the desired tradeoff between energy E and delay t. (Usually, $n = 2$.)

The only important aspect of the design flow that has not addressed in Phase I is *testing.* Although some members of the Situs team have contributed the first results on testing of asynchronous circuits, it would be convenient to leave the development of a systematic testing method to another group. However, an integrated approach as the one proposed by Situs may require that Situs develop their own testing procedure.

Situs will also be an early industrial user of the tools since we intend to be a design house as well as an EDA vendor. It has been awarded an SBIR Phase I contract to investigate the potential radiation-hardness advantages of asynchronous VLSI, which will require the design of experimental systems. Situs will also collaborate with Caltech, Cornell, and ISI, (and possibly other DoD prime contractors) inside the DARPA CLASS program. Situs will provide the EDA tools for the project and may be a subcontractor for some of the designs.

The marketing strategy will follow the well-known *technology adoption life cycle* defined for *discontinuous innovations.* The tools will first be offered to the "innovators" (or "technology enthusiasts") and the "early adopters" (or "visionaries")—at this point, it is difficult to differentiate the two categories of customers. Among them will be the async startups like Fulcrum Microsystems, small high-technology companies like Myricom and Tensilica, who know and appreciate the technology. One may also include in this class some defense contractors in the DARPA community and some small skunkworks groups inside large corporations. Such groups exist inside Intel, IBM, Sun, Apple, and others.

According to modern studies in the marketing of high technology, the difficult step is the next one, when we reach to the next category of customers called the "early majority" or "pragmatists." Situs hopes to turn the small insider groups within large corporations and the defense contractors into our early majority. In order to do so, it is necessary to have established Situs as the market leader in async tools by then.

3.5. Contract Delivery Status

On schedule.

3.6. Report Preparers

The report was prepared by Alain J. Martin, PI, Mika Nyström, and Catherine G. Wong.